Advanced Data Structures (COP 5536)
FALL 2016
Programming Project Report

CHETANA SEELAM
UFID:5115-3257
chetana@ufl.edu

PROJECT DESCRIPTION

Goal: To implement a system to find out the n most popular hashtags appeared on social media such as Facebook or Twitter, using advanced data structures.

Scope: For the scope of this project, we read the hashtags from an input file which may consist of more than one million hashtags.

Data Structures used: Max Fibonacci Heap and Hash Table are used to achieve the goal.

Max Fibonacci Heap -to keep track of frequencies of hashtags and to extract the maximum frequencies at any given time.

Hash Table- to maintain the key-value pairs where key is the hashtag and value is pointer to the corresponding node in the Max Fibonacci Heap.

WORKING ENVIRONMENT

Programming Language used: C++

The project has been compiled and executed on thunder.cise.ufl.edu, Code Blocks editor and xubuntu.

INSTRUCTIONS TO COMPILE

To compile the program,

type the command 'make', which makes use of the Makefile for compilation.

Sometimes this command gives missing separator error due to the missing tab space improperly inserted on different machines. In such cases, please insert a tab space in the Makefile, before giving the following commands:

```
$(CC) $(CFLAGS) -o $(TARGET) $(TARGET).cpp
$(RM) $(TARGET)
```

Or you can simply compile with the command:

```
g++ -g -fpermissive -o hashtagcounter hashtagcounter.cpp
```

To execute the program,

```
./hashtagcounter inputfilename
```

You can use the remotely accessible server using ssh i.e, thunder.cise.ufl.edu

You will see a prompt saying which file to refer for the output

PROGRAM STRUCTURE AND FUNCTION PROTOTYPES

I have used two generic classes FibonacciHeap and Node with the help of a class template T.

The class Node mainly describes the node structure, used in the Fibonacci heap. The class variables are:

degree: This variable is of type Integer and it signifies the number of nodes that a node can have in its next level.

chid_cut: This variable is of type Boolean and it signifies whether a child has already been removed from that node. A 'chid_cut' of false means that no child has ever been removed from that node.

The Node class has the pointers to the nodes parent, child, right and left. Right and left are used for the doubly linked list that is created at each level. All the nodes in the root list exist in a circular double linked list.

FibonacciHeap and Node classes have their own constructors and destructors.

Code Flow:

- The program starts its execution from the main function.
- It takes the input file and parses it. As the parser comes across the hashtags and their frequencies in the input file, they are inserted into a hash table and the Fibonacci heap at the same time.
- If a hashtag appears more than once, Increase Key () function is called, which has the functionality of computing the sum of the frequencies of that hashtag till then.
- If a line in input file contains only a numeric value(n), n most popular hashtags will be extracted from the Fibonacci heap and displayed. After this operation, the extracted nodes are immediately inserted into the heap so that they would help in extracting the popular hashtags in future.

Function Prototypes used:

void insert (Node *);

Parameters: Pointer to the Node to be inserted, which is of the Node class type

Return Type: void

This function inserts the nodes into the root list of the Fibonacci heap.

static FibonacciHeap* union_FibonacciHeap (FibonacciHeap *, FibonacciHeap *);

Parameters: Pointers to two Fibonacci heaps

Return Type: Pointer to the resultant heap

This function is used to concatenate two Fibonacci heaps.

Node* extract_max ();

Parameters: none

Return Type: Pointer to the node of Node class type

This function is used to extract the maximum node from the heap.

Void consolidate ();

Parameters: none

Return Type: void

This function is used to consolidate the heap as a part of extract_max() operation.

void fib_heap_link (Node*, Node*);

Parameters: Pointers to the two root nodes of the trees in the heap, that are to be linked.

Return Type: void

This function is used to link the roots (x & y) of two trees in a heap.

if key(x)<= key(y), we make x the child of y; otherwise, we make y the child of x. The appropriate node's degrees and appropriate child list are updated.

void increase_key (Node*, int);

Parameters: Pointer to the node whose value is to be increased and the int parameter to specify the numeric value by which the frequency is to be increased.

Return Type: void

This function is used to increase the frequency of a hashtag, stored in a particular node. Increase the value of node and if the value of node increased(x) is greater than its parent(y), cut the node and its children from the existing tree and insert into the root list of the heap. Chid_cut the node y to be 'true', and perform cascading_cut operations until we approach a node whose child cut value is false.

void cut (Node*, Node*);

Parameters: Pointers to the child node to be cut and pointer to its parent.

Return Type: void

This function is used to remove the node from the child list of its parent, and insert it into the root list of H, updating the appropriate variables.

void cascading cut (Node*);

Parameters: Pointer to the node on which cascading cut has to be performed

Return Type: void

This function is used to check the child cut values of nodes. If the child cut value is false for the parent, make it true. If the child cut is true, keep going up the tree and removing the nodes until it finds a node whose child cut is false.

void remove_Node (Node* x);

Parameters: Node to be removed.

Return Type: void

This function invokes increase key function and sets it to infinity, so that the node can be easily removed from the top of the heap.

void pop ();

This function makes use of extract_max () function and pops out the max frequencies.

Node* push (T, void *);

This function pushes the nodes into the heap.

bool isnum (char *);

This function returns a Boolean value by checking if the input parameter is a number or not.

void del_Nodes (Node *);

Parameters: Pointer to the node to be deleted.

Return Type: void

This function is used to delete the specified heap nodes

Conclusion

Thus the goal of extracting the top n hashtags from an input file ,given is accomplished using the max Fibonacci heap and the hashtable